课后作业四：决策树

任务：基于决策树构建机器学习模型，根据乘客的特征预测其在 Titanic 沉船事件中是否幸存。

数据集：Titanic 数据集中乘客的特征包含客舱等级、性别、年龄、在 Titanic 号上的同伴/配偶数量、船票编号、票价等。对于每一个乘客都包含了其是否在 Titanic 灾难中生还的信息（Survived），作为真实值标签。

普通实现：

1.导入必要的库

```
import graphviz  #一个开源的图形可视化工具，可以用来生成和展示图形结构
import pandas as pd
import pydotplus  #pydotplus 是一个图形库，基于 pydot 和 Graphviz，允许创建和操作 DOT 文件
from IPython.display import Image  #一个 IPython 模块，用于在 Jupyter Notebook 中嵌入和显示图像。
from sklearn.metrics import accuracy_score  #用于计算分类模型的准确率
from sklearn.model_selection import GridSearchCV, train_test_split  #优化模型的超参数
from sklearn.tree import DecisionTreeClassifier, export_graphviz  #用于创建和训练决策树分类模型
```

2.加载数据

```
# 加载数据
train_data = pd.read_csv('D:\\dataenclorse\\forth\\train.csv')
test_data = pd.read_csv('D:\\dataenclorse\\forth\\test.csv')
submission = pd.read_csv('D:\\dataenclorse\\forth\\submission.csv')
```

3.定义预处理函数

```
def preprocess_data(data):
    # 处理缺失
    data['Age'].fillna(data['Age'].median(), inplace=True)
    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)
    data['Fare'].fillna(data['Fare'].median(), inplace=True)

    # 将性别和登船港口转换为数值
    data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
    data['Embarked'] = data['Embarked'].map({'C': 0, 'Q': 1, 'S': 2})
    # 删除不必要的列
    data.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1, inplace=True)
    return data
```

4.数据预处理

```
# 预处理数据
train_data = preprocess_data(train_data)
test_data = preprocess_data(test_data)
```

```
# 特征和标签
X = train_data.drop('Survived', axis=1)
y = train_data['Survived']
# 分割数据集为训练集和验证集
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

5.定义参数网格

```
# 定义参数网格
param_grid = {
    'max_depth': [None, 10, 20, 30, 40, 50],
    'min_samples_split': [2, 10, 20],
    'min_samples_leaf': [1, 5, 10],
    'max_features': [None, 'sqrt', 'log2']
}
```

6.创建决策树分类器并验证准确率

```
# 创建决策树分类器
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# 验证模型
y_pred = clf.predict(X_val)
accuracy = accuracy_score(y_val, y_pred)
print(f'Validation Accuracy: {accuracy:.4f}')
```

7.对测试集进行预测并填入 submission.csv

```
# 对测试集进行预测
test_pred = clf.predict(test_data)

# 将预测结果填入 submission.csv
submission['Survived'] = test_pred
# 保存结果到 submission.csv
submission.to_csv('D:\\dataenclorse\\forth\\submission.csv', index=False)
可视化决策树并保存为图片
dot_data = export_graphviz(
    clf,
    out_file=None,
    feature_names=X.columns,
    class_names=['Not Survived', 'Survived'],
    filled=True,
    rounded=True,
    special_characters=True
)
```

8.可视化决策树

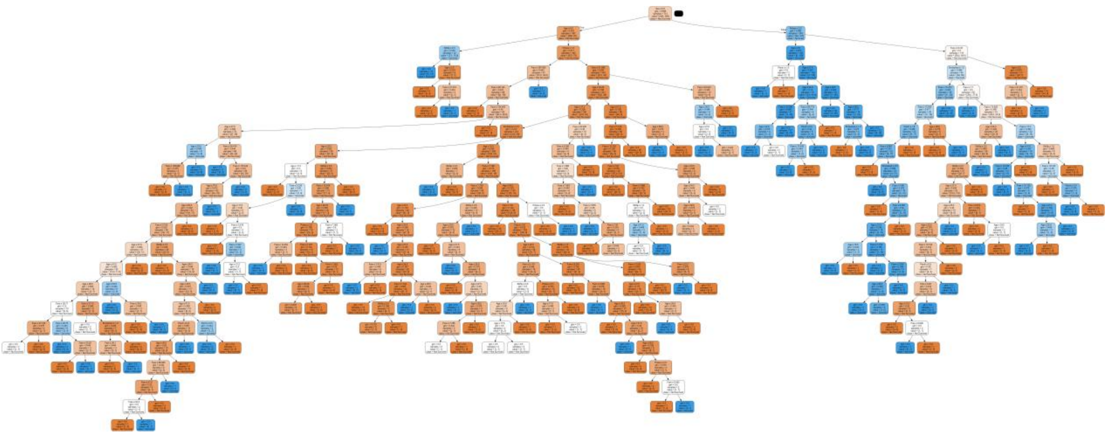
```
dot_data = export_graphviz(
    clf,
    out_file=None,
    feature_names=X.columns,
    class_names=['Not Survived', 'Survived'],
    filled=True,
    rounded=True,
    special_characters=True
)
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_png('D:\\dataenclorse\\forth\\decision_tree.png')
```

9.结果如下：

准确率为 0.7821，不是很理想

```
  data['Fare'].fillna(data['Fare'].median(), inplace=True)
Validation Accuracy: 0.7821
```

决策树如下（枝叶茂密，决策拖泥带水）：



优化实现：

1.使用网格搜索进行超参数调优

```
grid_search = GridSearchCV(clf, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
```

2.打印最佳参数和最佳得分

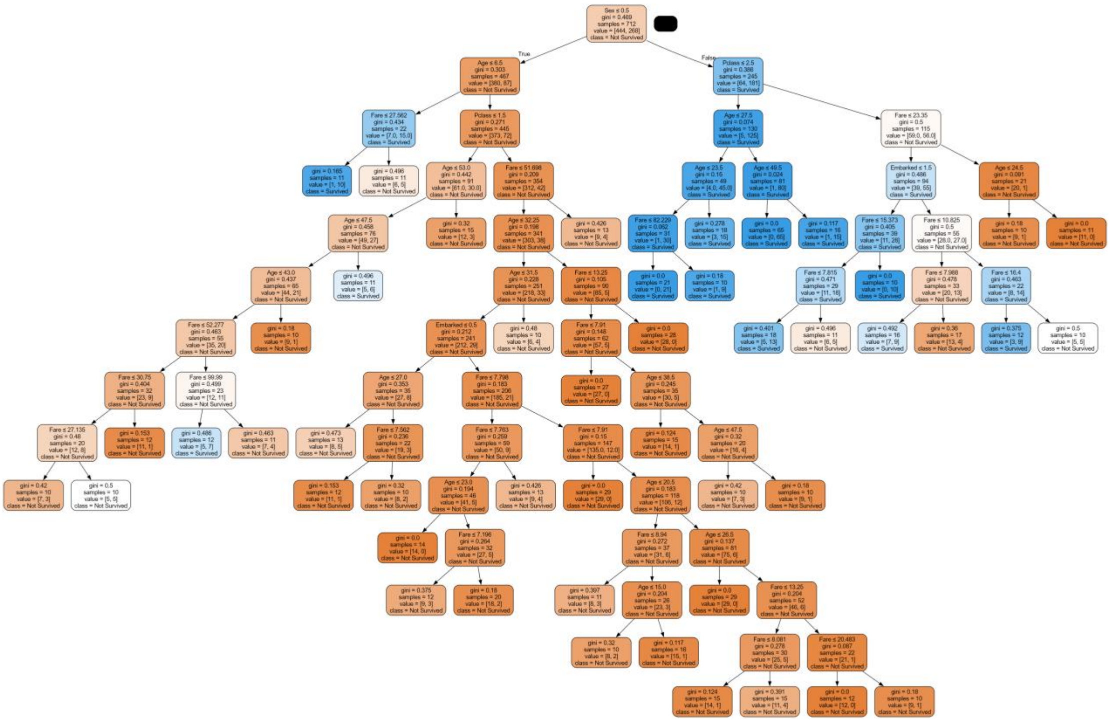
```
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print(f'Best Parameters: {best_params}')
print(f'Best Score: {best_score:.4f}')
```

3.优化结果如下：

最高分为：0.8104，准确率为 0.7989，有了部分提高

```
Best Parameters: {'max_depth': None, 'max_features': None, 'min_samples_leaf': 10, 'min_samples_split': 2}
Best Score: 0.8104
Validation Accuracy with Best Parameters: 0.7989
```

决策树如下（相比普通的决策树，优化后的决策树更加简洁高效）：



Submission.csv 如下（只展示部分）：

1	Passenger	Survived
2	892	1
3	893	0
4	894	1
5	895	0
6	896	0
7	897	0
8	898	0
9	899	0
10	900	0
11	901	0
12	902	0
13	903	0
14	904	1
15	905	1
16	906	0
17	907	1
18	908	0
19	909	1
20	910	0
21	911	1
22	912	1
23	913	0
24	914	0
25	915	0
26	916	1
27	917	0
28	918	0
29	919	1
30	920	0